Data Mining Capstone
Task 6

Best F1 score achieved : 0.572078065996
Rank on the leaderboard : 5
Alias : Andúril

I used python scripts for this task and my toolkit included using **nltk**, **pybrain** and **scikit-learn** for my models. All the scripts I used are available on my github profile at https://github.com/anuragprateek/yelp-yelp/tree/master/Task6.

Preprocessing:
I preprocessed the hygiene.dat text file using nltk by removing english stop words and punctuation from the text. I also converted all words to lower case. Removing stop words and punctuation removes noise from the data and helps create better models.

Text Representation:
I used 2 text representation techniques:

1. I used the Google word2vec[2] tool to obtain vector representation of words in the entire corpus. The word2vec tool takes as input a training text file, the size of the word vectors needed and outputs a file with the each line having one word from the corpus vocabulary along with its vector representation. I used word vectors of size 100 for my purposes.

2. I also tried using a document term frequency representation, where each line in the hygiene.dat file was represented as a vector with the exact number of times each word in the corpus vocabulary appeared in that review (by 'corpus vocabulary', I mean the collection of all the words that make up the training and test data, in this case the hygiene.dat file). But this sparse matrix representation is known to be not as effective as the word2vec representation since word2vec uses a continuous bag-of-words model to better simulate correlations between words. My best results were obtained by using the word2vec representation.

Learning Algorithms:
Each of the points below describes the learning algorithm used and the results obtained. I used 90 percent (used a larger percentage because of very small training set) of the training data to train my models and used the other 10 percent as my validation set.

1. Learning algorithm: **Multilayer Perceptron[1] (MLP)**
   Using the word2vec representation obtained, I used a bag-of-words model to train a multilayer perceptron (which is just an artificial feedforward fully-connected neural network). The bag-of-words model means that for any review, I took the average of the vector representations of all the words in that review. Because of this, the input layer of my MLP had 100 neurons (since each of my word vectors are of size 100), the hidden layer of my MLP had 50 neurons and the output layer had 2 neurons (one for each class 0 and 1. 0 stands for a restaurant passing the public health inspection and 1 stands for a restaurant failing the public health inspection). The output layer was a Softmax Layer[5] and produced a probability for either output (0 or 1). I took whichever probablity was larger to be the predicted class. To start off, I did not use any additional features as I wanted to set a benchmark. After training this MLP for about 200 epochs and submitting the obtained predictions on the test set, I got a F1 score of 0.557220694554. I mainly used pybrain[3] to build my MLP model.

2. Learning algorithm: **Logistic Regression**
   I then proceeded to check how a simple logistic regression module would perform.

Using the word vectors obtained from word2vec and LogisticRegression from scikit-learn[4] and without using any extra features, I was also able to obtain a decent F1 score of 0.555411288971. The logistic regression model performed slightly less well than the MLP. This could be because MLPs are known to learn better features from data, but this could also be just noise as the training data set really is too small. However, up to this point I hadn't even used my additional features yet or an ensemble of models. So that is what I did next!

3. Learning algorithm: **Mutlilayer Perceptron + Logistic Regression + Additional features**
   From the hygiene.dat.additional file. I retrieved the average rating of a restaurant, the number of ratings for a restaurant and the cuisines offered by a restaurant to use as additional features. I used my first MLP model (from point 1) to obtain probablities for the two classes 0 and 1. To get started, I just used these two probabilites along with just the average rating of each restaurant as features to train a logistic regression model. I got an improved F1 score of 0.569604093735. Kindly note that here I was **training my logistic regression model on top of my MLP model!**

   I then added the number of reviews and the cuisines offered as features as well. I used these 5 features (probablity of class 0 obtained from MLP, probability of class 1 obtained from MLP, average rating, number of reviews, and a set of cuisines offered) to train a logistic regression model. I was expecting improvements as I had used additional features and an ensemble of models and that is what I got (an F1 score of 0.572078065996, my best score yet.)

4. I also experimented with **Support Vector Machines** and **Random Forest** classifiers from the scikit-learn module. Both seemed to be giving comparable results but the logistic regresssion F1 score on the test set was slightly greater and hence I used it for my final model.

Ananlysis of results:
Using word2vec gave better text representation as it uses a continuous bag-of-words model trained using an artificial neural network that captures relationships between words quite closely. Using my MLP predictions and then training them using a logistic regression layer with a set of other features captured the best of all worlds (my MLP model had been trained on word vectors, and many false predictions by the MLP model were corrected when combined with other additional features in the logistic regression model).

I checked the predictions on the test data as well as the training data already provided and I guess the reason why nobody has been able to get a very high F1 score is because there is not much signal in the data. If you look at the training data, you'll find restaurants which have all positive reviews but have been classified as failing their public health inspection. I also wish the task had a validation set and that the test set was secret as that would have given a better measure of which models performed better on unknown data.

References

1. Multilayer Perceptron (https://en.wikipedia.org/wiki/Multilayer_perceptron)
2. Word2vec (https://code.google.com/p/word2vec/)
3. Pybrain (www.pybrain.org)
4. Scikit-learn (http://scikit-learn.org/stable/)
5. Softmax function (https://en.wikipedia.org/wiki/Softmax_function)
6. LogisticRegression in scikit (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)
7. SVM in scikit (http://scikit-learn.org/stable/modules/svm.html)
8. RandomForest in scikit (http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)